Trajectory Prediction of Traffic Agents: Incorporating context into machine learning approaches

Vyshakh Palli-Thazha, David Filliat and Javier Ibañez-Guzmán

***Abstract*— For a vehicle to navigate autonomously, it needs to perceive its surroundings and estimate the future state of the relevant traffic-agents with which it might interact as it navigates across public road networks. Predicting the future state of the perceived entities is a challenge, as these might appear to move in a stochastic manner. However, their motion is constrained to an extent by context, in particular the road network structure. Conventional machine learning methods are mainly trained using data from the perceived entities without considering roads, as a result trajectory prediction is difficult. In this paper, the notion of maps representing the road structure are included into the machine learning process. For this purpose, 3D LiDAR points and maps in the form of binary masks are used. These are used on a recurrent artificial neural network, the LSTM encoder-decoder based architecture to predict the motion of the interacting traffic agents. A comparison between the proposed solution with one that is only sensor driven (LiDAR) is included. For this purpose, NuScenes dataset is utilised, that includes annotated 3D point clouds. The results have demonstrated the importance of context to enhance our prediction performance as well as the capability of our machine learning framework to incorporate map information.**

I. Introduction

Sequence prediction problems have been an important area of research with multiple applications like Natural Language Processing, Weather Forecast, Economics and Intelligent Systems. Autonomous vehicle systems come across this problem when trying to understand the behaviour of all other traffic-agents that it interacts with. This could be a pedestrian, car, bike or any object that appear in its navigation space. Predicting the trajectory of the involved agents allow the autonomous vehicle to prepare itself for future decisions and manoeuvres, after assessing the risk involved in the current scenario. The goal is to try and make the vehicle understand the environment like humans do. The high level of interactions, occlusions and changing context makes it hard to solve this prediction problem with high accuracy. Therefore multiple inputs, patterns and behaviours must be studied to come to an operational and acceptable accurate prediction.

Researchers have addressed the problem focusing on different aspects. Sequence predictions for single-agent behaviours have been solved using classical techniques but in real life this is unusually the case. There are multiple agents and their behaviour changes according to the class of the agent - whether it is a car, a pedestrian or a bike. Our research takes the class of the agent into account in the problem statement and tries to predict their different behaviours. Another factor that plays an important role in real life traffic scenarios are the interactions between different classes of agents. Our current research also gives importance to this and has incorporated an interaction-aware model. Thirdly, our model takes into account, the surroundings of the ego-motion car, by using a map-mask that provides information on the driveable and non-driveable areas for different classes of agents. These map-masks also provide information on the path that could be taken by different agents.

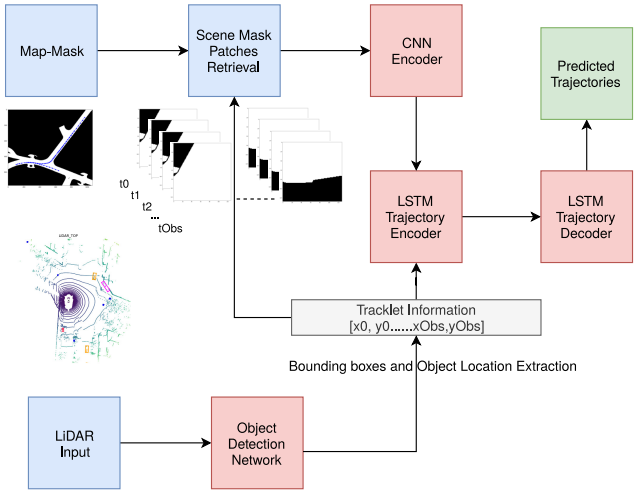


Fig. 1: System Architecture: The prediction module has two inputs: sequential historical trajectories of all traffic-agents and the associated map-mask patches of each agent for the observed time instances

In the recent years, the boom in high-computational-power processors and GPUs has lead to faster research and development cycles. More and more data-sets are being released for different applications - KITTI, Apolloscape, NuScenes and NGSIM. This has lead to a focus on deep learning algorithms which solve classical problems with extremely high rate of success and accuracy. This success of Learning Algorithms have attracted research on several methods like Convolutional Neural Networks(CNN), Recurrent Neural Networks(RNN), Long Short-Term Memory(LSTM) and Generative Adver-

Vyshakh Palli-Thazha is a PhD candidate at ENSTA Paris and Groupe Renault, Paris, France vyshakh.palli-thazha@ensta-paris.fr, vyshakh.v.palli-thazha@renault.com

David Filliat, Director, U2IS Lab, ENSTA Paris, Institut Polytechnique de Paris david.filliat@ensta-paris.fr

Javier Ibañez-Guzmán, Expert, Artificial Intelligence and Autonomous Systems, Groupe-Renault javier.ibanez-guzman@renault.com

sarial Networks(GANs). RNNs and LSTMs have shown promising results with solving sequence prediction problems. We use one such LSTM encoder-decoder architecture (see Figure 3 for an overview) as the baseline for our study and use the NuScenes dataset [1] which provides 3D bounding-boxes in LiDAR point-clouds for different traffic agents, object/agent IDs and do a preliminary study for improving trajectory prediction using map-masks (binary maps) for the environment which we use in our experiments.

In this work, the use of map-mask patches to improve the prediction of trajectories for different classes of interacting traffic-agents is proposed. Specifically, our contributions include:

- A new LSTM encoder-decoder architecture that uses Map-Mask patches to make trajectory predictions for different classes of traffic agents in drivable and non-drivable areas
- We evaluate the proposed model in comparison with LSTM baselines and is found to be superior in performance both in single-agent and multi-agent interaction scenarios.

The remainder of the paper is organised as follows: Section II mentions relevant research papers and results that we have taken inspiration from or has guided our research. Research related to single-agent predictions, multi-agent predictions, interaction-aware models, context-aware models are discussed in this section. Section III describes our map-mask based LSTM encoder-decoder architecture and the working of our complete trajectory prediction approach. Section IV discusses the results and the performance of our model and finally section V concludes the paper and mentions the future work.

II. Related Works

A. Classical Methods

Trajectory prediction for pedestrians, vehicles and other traffic agents has been studied using classical techniques such as Constant Velocity, Constant Acceleration, Linear Regressions, Kalman Filters [2], Monte Carlo Simulation [3], Time-series methods and Hidden Markov Models [4]. Most of these predictions are limited to short term predictions and even if these are long-term, their accuracy is affected because of the lack of context information and from ignoring the interaction between different classes of traffic-agents. A survey on motion prediction can be found in [5] where Lefèfre *et. al.* have analysed different trajectory prediction methods based on model completeness and risk assessment.

B. Learning based Models

Trajectory prediction methods have been applied to different sensor outputs, some solely with vision based detection outputs, others with 3D bounding-boxes obtained from LiDAR point-clouds. We base our work with the detections on 3D point-cloud data. Much research has gone into developing deep learning algorithms to infer such predictions. Altché *et. al.* [6] uses LSTMs to predict the future trajectory of a single target vehicle. It predicts the longitudinal and lateral trajectories of vehicles on a highway. In [7] A. Milan *et. al.* proposed an end-to-end learning approach for online multi-target tracking which uses RNNs to predict the state of each target step by step and uses LSTMs to achieve data association. They work on vision data obtained from the 3D MOTChallenge dataset. Nikhil *et. al.* [8] uses a Convolutional Neural Network based approach to predict trajectories of surrounding vehicles. This model utilises highly parallelisable convolutional layers to handle temporal dependencies. The trajectory histories are embedded to a fixed size tensor and stacked convolutional layers are used to ensure temporal consistency. Jawed *et. al.* [9] also uses a similar architecture based on convolutional layers.

C. Interaction and/or Context Aware Models

There are several examples in the literature that tries to solve the prediction problem which involves interactions between traffic agents. Some of them focus on pedestrians alone or vehicles alone while others try to solve cases with interaction between multiple classes of agents.

Several models study interaction between vehicles: In [10] Hu *et. al.* uses a generative model to jointly predict the sequential motions of each pair of interacting vehicles and it uses as input past trajectories and environment information. Kim *et. al.* [11] uses occupancy grid and RNNs to predict the trajectory of vehicles on highways. In [12] Deo *et. al.* uses an an LSTM encoder-decoder architecture that uses convolutional social pooling for learning inter-dependencies between vehicles. Here the spatial configuration of the agents in a scene is embedded into a spatial grid around the ego-motion vehicle and this grid is passed into convolutional and pooling layers to obtain the social context encoding. This along with an LSTM encoding of the agents' trajectory are concatenated and passed through a decoder to obtain the final predicted trajectory. A similar architecture is used by Park *et. al.* [13] with the decoder producing K likely trajectories over an occupancy grid using the beam search technique.

Some models study pedestrian interactions: Kooij *et. al.* [14] proposes a Dynamic Bayesian Network which uses the situational awareness and spatial layout to predict pedestrian paths. It uses pedestrian head orientation, distance between vehicles and pedestrians at expected point of closest approach and distance of pedestrian to curbside as latent states on a Switching Linear Dynamical System (SLDS) to predict changes in pedestrian dynamics. In [15] Alahi *et. al.* tries to learn general human movement and predicts their future trajectories. They model pedestrian motion by a pooling based LSTM architecture and jointly predicts the trajectories of all the people in a scene.

Models also use various context information to improve trajectory prediction: Lee *et. al.* [16] applies a Conditional Variational AutoEncoder (CVAE) based RNN encoder-decoder to make prediction for interacting agents. They use past trajectories as input along with a feature map, generated from scene elements like roads and sidewalks, that provides context information to the prediction model. Habibi *et. al.* [17] uses relative distance to curbside and status of pedestrian

traffic lights as additional information to provide context to predict the pedestrian path.

Our approach addresses the problem by using map-masks to facilitate the understanding of the scene environment. This study is a first step towards using semantic information in the form of maps for improving trajectory prediction. The trend is to add more information to maps. Works like OpenStreetMaps and NuScenes are releasing maps with information other than road geometry. Details such as cross-walk, sidewalk, traffic lights, stop lines, lanes are being added to maps. These semantics could be used to improve the perception of the environment.

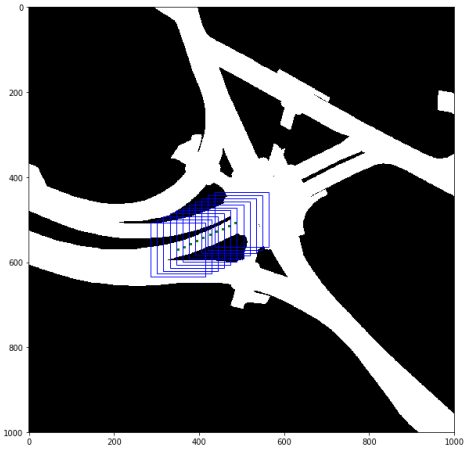
III. Problem Formulation and Method

In a driving scenario, several objects might be present, sharing the same ego-vehicle work-space it navigates. These include pedestrians, other vehicles or cyclists. State-of-the-art perception algorithms provide bounding boxes/segmentation for each class of objects. There are algorithms which can perceive the said objects from camera images or from LiDAR point-cloud data. In our case, we use LiDAR point-cloud data and off-the-shelf detection algorithms to extract the features of all the traffic-agents within the field of view of the sensor used. For experimental purposes, the PointPillars object-detection network by Lang *et. al.* [18] is adopted. This network is chosen as it shows superior performances over other relevant frameworks and they have released their code on GitHub. The observed trajectory is taken from the output of the object detection framework. For training the prediction network, the ground truth trajectory available for the situation from the NuScenes dataset is used.

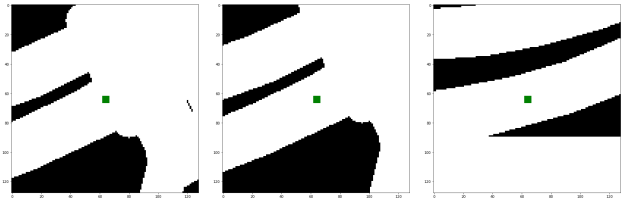
The feature set of each traffic agent k at time t can be defined by $f_t^k = (p_t^k, c_t^k, M_t^k)$ where $p_t^k = [x_t^k, y_t^k]$, x and y are the spatial coordinates of the detected traffic-agent and c is the class of the traffic agent, $c_i \in (1, 2, 3)$ where 1 is for pedestrians, 2 is for cars and 3 is for cyclists and M_t^k is a 128*128 pixels binary map patch of the area surrounding each detected object. The task is to observe the features of all traffic-agents in the interval $[1 : T_{obs}]$ and predict their positions in $[T_{obs+1} : T_{pred}]$.

A. Trajectory Prediction using LSTMs

The prediction framework developed by T. Fernando *et. al.* [19] that applies LSTMs for predicting trajectories is applied, along with the training methods adopted by Bahdanau *et. al.* in their Neural Machine Translation work [20]. For each time-step, an object-pool is created, associated with a trajectory prediction module. It regresses short term and long-term trajectories for each traffic-agent in the object-pool. This is based on the historic trajectory of that agent and neighbouring trajectories. Therefore, each traffic-agent that resides in the pool is associated with a probable short term and long-term trajectory. This approach is advantageous because raw detection outputs can be noisy due to occlusions, false alarms, inaccurate bounding boxes, and missing detections. Studies show that occlusions last around 3-10 frames [21].



(a) Binary map-mask of a scene



(b) Examples of patches centred around the agent

Fig. 2: Binary map-mask and map patches: A trajectory and the map-patches surrounding the agent are marked by blue squares, centred around the traffic agent marked in green

Trajectory prediction:
Let the historical trajectory of a traffic-agent *k*, from time-step 1 to time-step T_{obs} be given by,

$$p^k = [p_1, ..., p_{T_{obs}}],$$

where,

$$p_i = [x_i, y_i]$$

for each time-step *i*. p_i composes of points in a Cartesian grid obtained from the top-down view of the LiDAR point-cloud detections. These historical trajectories are passed through the LSTM encoder of each respective traffic-agent to generate its historical embeddings as follows,

$$h_t^k = LSTM(p_t^k, h_{t-1}^k),$$

generating a sequence of historical embeddings.

A historical context vector $C_t^{H,k}$ is defined to encode the trajectory information from the traffic-agent of interest (k), which can be computed as a weighted sum of hidden states from $t = [1 : T_{obs}]$,

$$C_t^{H,k} = \sum_{j=1}^{T_{obs}} \alpha_{tj} h_j^k$$

and the weight α_{tj} as shown in [20] can be computed by,

$$\alpha_{tj} = \frac{exp(e_{tj})}{\sum_{l=1}^{T} exp(e_{tl})}$$

$$e_{tj} = a(h^k_{t-1}, h^k_j)$$

where the function a is a feed forward neural network which is trained jointly with the whole network.

The spatial context vector $C^{S,k}$ is used for embedding the neighbouring trajectories.
The spatial weights, denoted by w^n_j, can be computed as,

$$w^n_j = \frac{1}{dist(n,j)}$$

where $dist(n,j)$ is the Euclidean distance between the n^{th} neighbour and the traffic-agent of interest at the j^{th} time-step, and w^n_j is the generated spatial attention weight. When there are N neighbouring trajectories in the local neighbourhood, and h^n_j is the encoded hidden state of the n^{th} neighbour at the j^{th} time-step, then the context vector for the spatial model is defined as,

$$C^{S,k} = \sum_{n=1}^{N} \sum_{j=1}^{T_{obs}} w^n_j h^n_j$$

The merged context vector, $C^{*,k}_t$, computed by,

$$C^{*,k} = tanh([C^{H,k}; C^{S,k}])$$

is then passed through the LSTM decoder to predict the future trajectory for the traffic-agent of interest. This is explained in Section III-C.

It is by modifying the context vector that the information from the binary map-masks is input to the prediction model. This is explained in the next sub-section.

B. Map Input for Trajectory Prediction

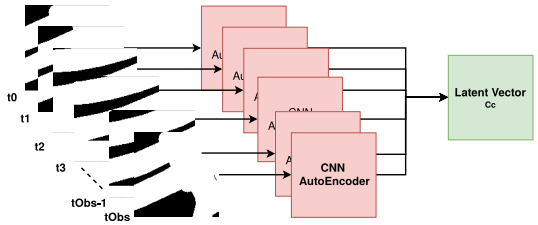


Fig. 3: Passing the map-patches through a CNN AutoEncoder generates the latent vector that stores information about the drivable area and this vector is passed on to the LSTM encoder architecture

For understanding the behaviour of different classes of traffic-agents in drivable and non-drivable spaces, the input of map information in the form of binary patches is studied. We take square patches of the size 128*128 pixels (see Figure 2) around each detected traffic-agent in the perception space (see Figure 2a) of the ego-motion vehicle and for each time-step in the observed time interval. Each of these patches are passed through a Convolutional AutoEncoder producing a latent vector C^C_t which holds the information about the drivable space associated with the respective traffic-agent (see Figure 3).

This vector is then concatenated with the merged context vector C^*_t from the LSTM Model to obtain the new latent vector $C^{M,k}_t$.

C. LSTM Decoder

Now, the final predicted trajectory q_t is obtained by passing the context vector $C^{M,k}_t$ in the case of map input or $C^{*,k}_t$ in the case of only LSTMs, through the LSTM decoder and two fully connected layers. The overall system architecture is illustrated in Figure 1.

$$q_t = LSTM_c(h^k_{t-1}, q_{t-1}, C^{M/*,k}_t)$$

q^k_t is composed of points in a Cartesian grid. The decoder used for each traffic-agent class is different. Hence, based on c, we choose the LSTM to be used to decode. We let $T_{obs} = 3$ for short term predictions and $T_{obs} = 10$ for long term trajectory predictions.

$$q_t = [x_{Obs+1}, y_{Obs+1}, ..., x_{Obs+N}, y_{Obs+N}]$$

where N is the prediction horizon.

The CNN AutoEncoder is tested both with and without pre-training and no significant difference has been found. Therefore, this model is attached to the LSTM encoder-decoder and trained end-to-end with Adam optimiser and a learning rate of $1 \times e^{-4}$ for 100 epochs and fine-tuned with a learning rate of $1 \times e{-}5$.

IV. Results and Evaluation

The prediction accuracy is measured based on two criteria as described in [22], [23]: Average Displacement Error (ADE) and Final Displacement Error (FDE) in pixels with respect to each time step t within the prediction horizon:

1) Average Displacement Error (ADE): Average distance between ground truth and our prediction over all predicted time-steps for the traffic agent.

$$ADE(i) = \frac{1}{T} \sum_{j=1,2...T} \sqrt{(\hat{x}^j_i - x^j_i)^2 - (\hat{y}^j_i - y^j_i)^2}$$

$$ADE = \frac{1}{n} \sum_{i=1,2,...n} ADE(i)$$

2) Final Displacement Error (FDE): The distance between the predicted final destination and the true final position of the agent.

$$FDE(i) = \frac{1}{n} \sqrt{(\hat{x}^T_i - x^T_i)^2 - (\hat{y}^T_i - y^T_i)^2}$$

$$FDE = \frac{1}{n} \sum_{i=1,2,...n} FDE(i)$$

where n is the total number of interacting agents in the test set, x_{ij} and y_{ij} denote the coordinates of the i^{th} agent in the predicted time-step j and T denotes the final predicted time-step.

As training data, the NuScenes dataset is used. From this we obtain 3200 different trajectories with enough length (at least 30 time frames) for the experiments. These trajectories

are obtained from 850 scenes of 20s recordings from the NuScenes data-collection car setup. These are split into train, validation and test sets of size 2100, 550 and 550 trajectories respectively. For single-agent trajectory prediction, these sequences are considered separately, and for the multi-agent trajectory predictions, the information of all interacting agents are taken into account. Each trajectory sequence is associated with the relevant interacting agent information also.

In order to illustrate the interest of using drivable/non-drivable map information, four variants of our approach are implemented.

- **Constant Velocity Model:** Trajectory predicted by assuming constant velocity to that of the observed trajectory. [24]
- **Constant Curvature Model:** Trajectory predicted by assuming constant curvature to that of the observed trajectory. [25]
- **Single Agent LSTM Model for all traffic agents:** Separate LSTM models are used to predict the motion of single traffic agents. No interaction is involved.
- **Single Agent with LSTM Model and Map-Mask input:** Map-mask is taken as input to predict the trajectories of single agents, not considering interactions.
- **Multi-Agent interactions and Model with LSTM only:** Interactions are taken into consideration and only LSTM encoder-decoder architecture is used. This is similar to the method in [26].
- **Multi-Agent interactions and Model with LSTM and Map-Mask input:** Multi-agent interactions are taken into account in the LSTM encoder-decoder architecture and map-masks are used to improve the predictions of traffic-agents involved in such interactions.

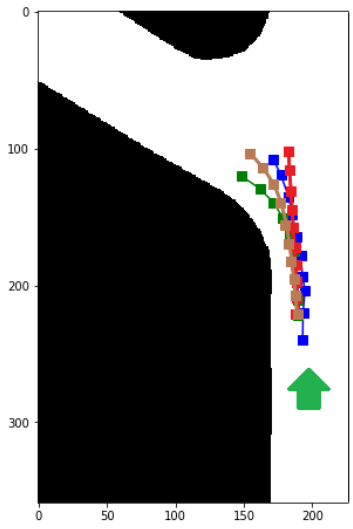


Fig. 4: Comparison of trajectory prediction with classical methods. Green is ground truth, red is constant velocity, brown is constant curvature and blue is map-mask based prediction

Method	ADE(Meters)	FDE(Meters)
1. Constant Velocity Model	7.65	8.2
2. Constant Curvature Model	5.47	6.78
3. Single Agent LSTM only	5.31	6.2
4. Single Agent LSTM + Map-Input	**1.67**	**1.95**
5. Multi-Agent LSTM only	4.34	5.21
6. Multi-Agent LSTM + Map-Input	**1.45**	**2.2**

TABLE I: Comparison of Prediction Accuracy for different models

The performance of the model in different scenarios is shown in Figure 4 and 5. Figure 4 shows the comparison of our models with classical approaches like the constant velocity model and constant curvature model. Although the constant curvature model shows a better performance in the particular situation depicted in Figure 4, the overall performance of the proposed model exceeds that of the constant curvature model. Figure 5a shows an example of straight trajectories and the prediction giving results in the non-drivable area when only LSTMs are used and better predictions confined to the drivable area when map-mask is added as input. Also the fitting of the trajectory prediction to road curves is better predicted when map-masks are introduced as can be seen in 5b. Another example of prediction being made in the drivable area is seen in Figure 5c.

The comparison of the ADE/FDE values for the LSTM baselines and the models with map-mask input is presented in Table I.

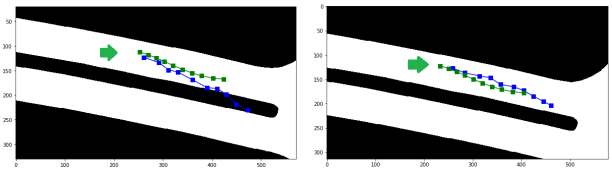
As can be seen in each case, single agent and multi-agent predictions, the addition of the map-mask patches help the system make more accurate predictions. This gives us confidence to study the use of a map with much more information in the future to improve the predictions even more.

V. Conclusion and Future Work

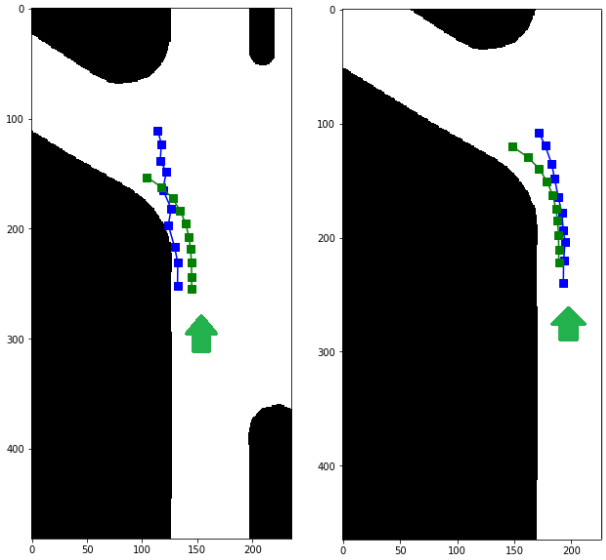
Trajectory prediction for interacting traffic agents is a major part of autonomous vehicle navigation and the proposed approach successfully improves upon classical and sensor only methods. It uses the knowledge of the road network structure in the form of binary masks in conjunction with 3D LiDAR points to provide vital information about the driving scenario. A comparison of single-agent and multi-agent trajectory prediction models is analysed and the overall improvement of the prediction when map-masks are given as input is asserted. This study serves as a preliminary step in the direction of using environmental context information for improving prediction accuracy. As a future work, we plan to study the Map Extension tool of the NuScenes dataset and use the multitude of information that can be obtained from their wide network of mapped areas. We also plan to use images to provide human like perception of the driving scene.

References

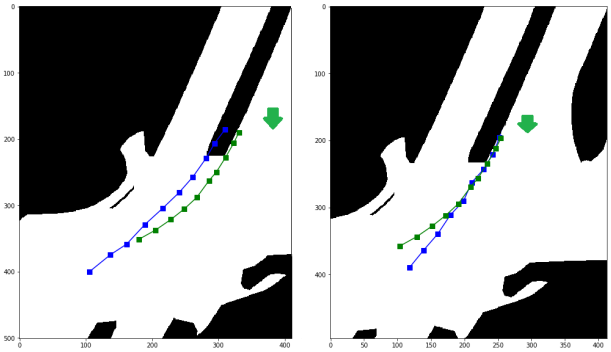
[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, “nuscenes: A multimodal



(a) Straight path trajectory predictions



(b) Trajectory prediction at a turnings



(c) Better behaviour in non-drivable areas

Fig. 5: Behaviour of Trajectory Prediction Models without(Left) and with Map-mask(Right) input. The green arrows in the images show the direction of prediction. The green path is the Ground truth and the blue path is the predicted trajectory

dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[2] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, 1960.

[3] S. Danielsson, L. Petersson, and A. Eidehall, "Monte carlo based threat assessment: Analysis and improvements," 07 2007, pp. 233 – 238.

[4] J. Firl, H. Stubing, S. Huss, and C. Stiller, "Predictive maneuver evaluation for enhancement of car-to-x mobility data," 06 2012, pp. 558–564.

[5] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, p. 1, 2014. [Online]. Available: https://hal.inria.fr/hal-01053736

[6] F. Altché and A. de La Fortelle, "An LSTM network for highway trajectory prediction," *CoRR*, vol. abs/1801.07962, 2018. [Online]. Available: http://arxiv.org/abs/1801.07962

[7] A. Milan, S. H. Rezatofighi, A. R. Dick, K. Schindler, and I. D. Reid, "Online multi-target tracking using recurrent neural networks," *CoRR*, vol. abs/1604.03635, 2016. [Online]. Available: http://arxiv.org/abs/1604.03635

[8] N. Nikhil and B. T. Morris, "Convolutional neural network for trajectory prediction," *Computer Vision - ECCV 2018 Workshops*, pp. 186–196, 2019. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-11015-4_16

[9] S. Jawed, E. Boumaiza, J. Grabocka, and L. Schmidt-Thieme, "Data-driven vehicle trajectory forecasting," 2019.

[10] Y. Hu, W. Zhan, and M. Tomizuka, "Multi-modal probabilistic prediction of interactive behavior via an interpretable model," *CoRR*, vol. abs/1903.09381, 2019. [Online]. Available: http://arxiv.org/abs/1903.09381

[11] B. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/itsc.2017.8317943

[12] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/cvprw.2018.00196

[13] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/ivs.2018.8500658

[14] J. F. P. Kooij, N. Schneider, F. Flohr, and D. M. Gavrila, "Context-based pedestrian path prediction," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 618–633.

[15] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, F. F. Li, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," 06 2016, pp. 961–971.

[16] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.233

[17] G. Habibi, N. Jaipuria, and J. P. How, "Context-aware pedestrian motion prediction in urban intersections," 2018.

[18] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," 2018.

[19] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft + hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection," *Neural Networks*, vol. 108, p. 466–478, Dec 2018. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2018.09.002

[20] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[21] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.41

[22] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2018.00240

[23] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[24] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," 2019.

[25] J. Horst and A. Barbera, "Trajectory generation for an on-road autonomous vehicle - art. no. 62302j," *Proc SPIE*, pp. 82–, 06 2006.

[26] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Tracking by prediction: A deep generative model for mutli-person localisation and tracking," *CoRR*, vol. abs/1803.03347, 2018. [Online]. Available: http://arxiv.org/abs/1803.03347